

class Solution **{ //DFS**

public boolean isSymmetric**(**TreeNode root**)** **{**

**return** solver**(**root**,** root**);**

**}**

public boolean solver**(**TreeNode a**,** TreeNode b**){**

**if(**a **==** **null** **&&** b **==** **null)** **return** **true;**

**if(**a **==** **null** **||** b **==** **null)** **return** **false;**

**return** **(**a**.**val **==** b**.**val**)** **&&** solver**(**a**.**left**,** b**.**right**)** **&&** solver**(**a**.**right**,** b**.**left**);**

**};**

**}**

class Solution **{ // Iteratively**

public boolean isSymmetric**(**TreeNode root**)** **{**

Queue**<**TreeNode**>** q **=** **new** LinkedList**<>();**

q**.**add**(**root**);**

q**.**add**(**root**);**

**while(!**q**.**isEmpty**()){**

TreeNode a **=** q**.**poll**();**

TreeNode b **=** q**.**poll**();**

**if(**a **==** **null** **&&** b **==** **null** **)** **continue;**

**if(**a **==** **null** **||** b **==** **null** **)** **return** **false;**

**if(**a**.**val **!=** b**.**val**)** **return** **false;**

q**.**add**(**a**.**left**);**

q**.**add**(**b**.**right**);**

q**.**add**(**a**.**right**);**

q**.**add**(**b**.**left**);**

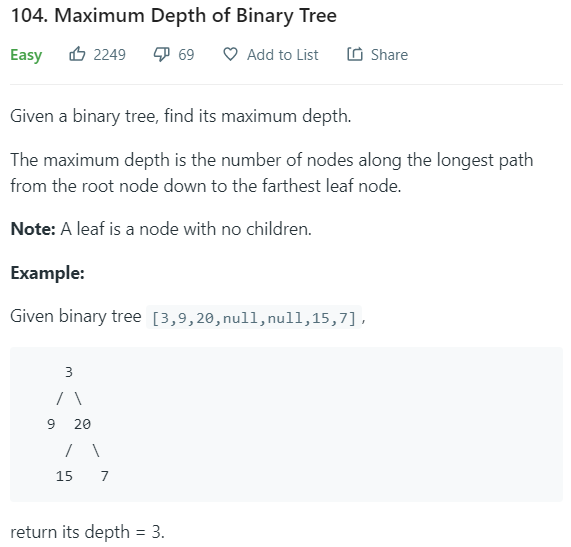
**}**

**return** **true;**

**}**

**}**

-------------------------------------------------------------------------------------



class Solution **{**

public int maxDepth**(**TreeNode root**)** **{**

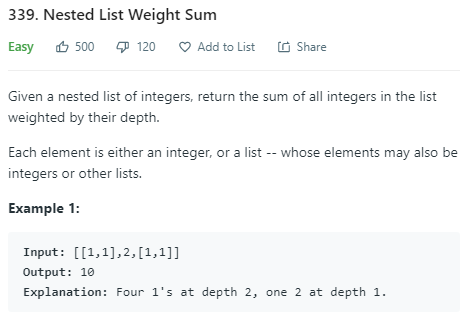
**if(**root **==** **null)** **return** 0**;**

**return** 1 **+** Math**.**max**(**maxDepth**(**root**.**left**),** maxDepth**(**root**.**right**));**

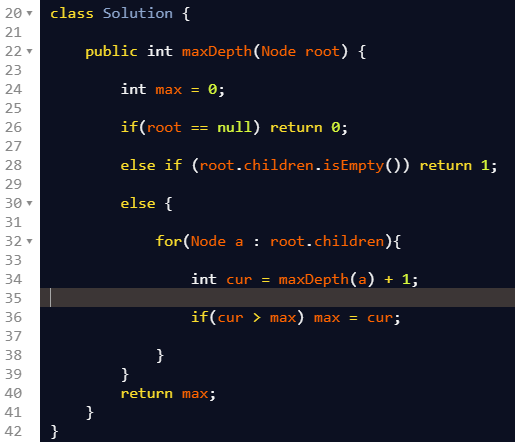
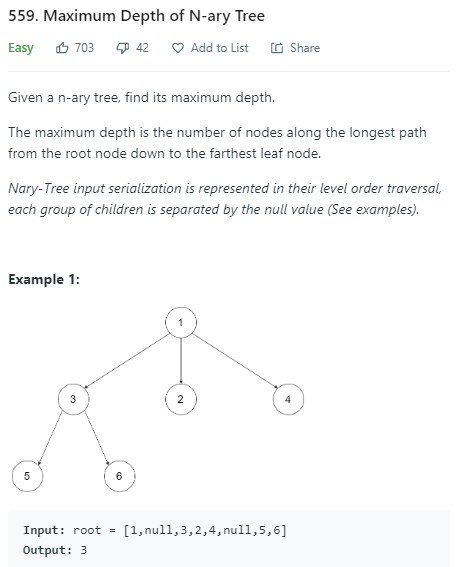
**}**

**}**

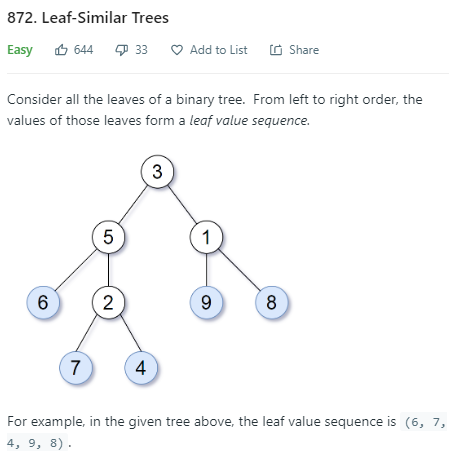
-------------------------------------------------------------------------------------

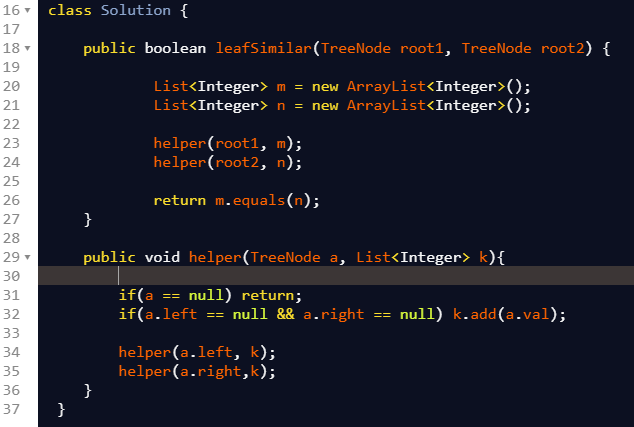


-------------------------------------------------------------------------------------

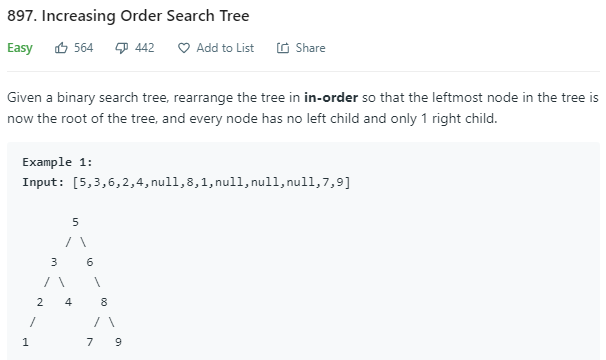


-------------------------------------------------------------------------------------





-------------------------------------------------------------------------------------



class Solution **{**

public TreeNode increasingBST**(**TreeNode root**)** **{**

List**<**Integer**>** vals **=** **new** ArrayList**();**

inorder**(**root**,** vals**);**

TreeNode ans **=** **new** TreeNode**(**0**),** cur **=** ans**;**

**for** **(**int v**:** vals**)** **{**

cur**.**right **=** **new** TreeNode**(**v**);**

cur **=** cur**.**right**;**

**}**

**return** ans**.**right**;**

**}**

public void inorder**(**TreeNode node**,** List**<**Integer**>** vals**)** **{**

**if** **(**node **==** **null)** **return;**

inorder**(**node**.**left**,** vals**);**

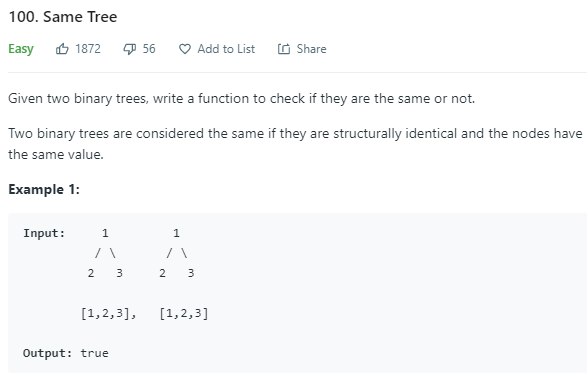
vals**.**add**(**node**.**val**);**

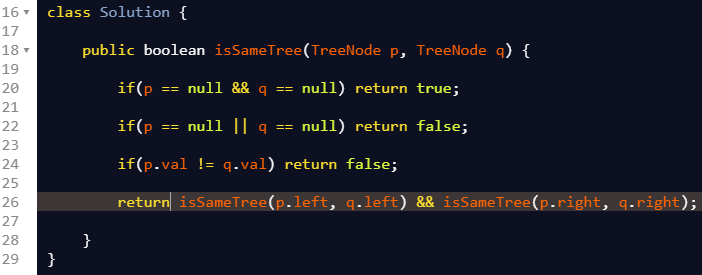
inorder**(**node**.**right**,** vals**);**

**}**

**}**

-------------------------------------------------------------------------------------





-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------